Back to Overview Stage 2

Stage 3

This stage of the project you will handle session state and will complete the implementation of the data component using the database access technology, **JDBC**. Also, **JSTL** will be used to permit the display of array information returned from the java bean and/or data access object components. Where needed, data is now saved to and retrieved from session attributes. Note that existing Filter and Control class changes are shown in green.

Important: Do not share your or your partner's java code with anyone.

This stage of the project will use the Limara database (if you are using Oracle, then use the Limara script). The database has three tables (plus a lookup table) which are described below. To get the specific details of primary keys, column types, constraints, and foreign keys, attach the database to Microsoft SQL Server Express and examine the tables in the designer. A brief description of the purpose of each table follows:

The AccountOwner Table

The accountowner table contains a list of all of the library account owners of the library materials reservation application. Each library account owner has an id (an identity column, so it is automatically generated), a first name, a last name, a username, and a password. You can manually add library account owners to this table as needed.

The Material Table

The material table contains the details of each of the library materials used by the library materials reservation application. These details include the material id, the library material title, and the type of the library material. The library material type must match one of the values in the material type lookup table (**MaterialTypes**). You can manually add library materials to this table as needed (some have already been included). To include more types, add new material types to the materiatypes table.

The MaterialReservation Table

The material reservation table contains the library material reservations made by the library account owners. The columns in this table include library material id which is a foreign key reference to a library material in the **Material Table**. Each library material reservation has a reservation expiration date & time. There is also a library account owner id column which is a foreign key reference to a library account owner in the **AccountOwner Table**. Each library material reservation uses the library material id to uniquely identify the library material reservation.

Special Note for Oracle Users

If you are using Oracle, you will need to make a minor adjustment to Part A. The change described below will allow your code to work with both SQL Server and Oracle simultaneously. Note that even if only one member of the group is using Oracle, this change will have to be made to Part A.

The code required to add a new record to the **AccountOwner** table must get the generated ID from the associated sequence. E.g., for the **AccountOwner** table, you would replace the line

```
ResultSet rsIdent = statement.executeQuery("SELECT @@IDENTITY");
```

with

```
String sql = ("SELECT @@IDENTITY");
if ("Oracle".equals(conn.getMetaData().getDatabaseProductName()))
   sql = "SELECT ACCOUNTOWNER_SEQUENCE.CURRVAL FROM DUAL";
ResultSet rsIdent = statement.executeQuery(sql);
```

This stage is worth **5%** of the 15% project mark. Submit both parts (Group A and Group B) into a single Eclipse Dynamic Web Project. Zip the entire project folder. Submit the project to the submission drop box by 11:59PM on **Tuesday December 1, 2015**. Presentations will be on Friday December 4, 2015. Be sure to include your name *within* the XHTML files that you author (as XHTML <-- comment --> tags) and as java comments (// comment) within your java source files. Note that the XHTML comments cannot be the first line in the .jspx file.

Group A Components

Members of this group are responsible for updating the classes **MaterialData** and **AccountOwnerData** to store and retrieve data from the database. The updated MVC **Control** classes for this group are also specified. The class updates are described below.

The `DataSourceFactory` Class

Create this class in a **data** sub-package under your project package.

Methods

`public static DataSource getDataSource()`
Returns the current **DataSource** object.

`public static void setDataSource(DataSource dataSource)`
Sets the current **DataSource** object.

The `MaterialData` Class

Methods

`public List<Material> getAvailableMaterials(String title, String type) throws SQLException` (see interface description)
Returns a list of **Material** objects representing all of the library materials in the database that are available for reservation according to the library material title or partial title and library material type requirements specified in the parameters. Any library materials that are currently reserved are not available.

If the title parameter is not **null**, then the method should find and return only library materials that contain the title or partial title specified. For example, if the title text is 'st', then library materials with the titles *Star Gazing* and *How to find a lost puppy* should be included in the list. If the title parameter is **null**, library materials with any title can be included in the returned results.

If the library material type parameter is not null, then the method should find and return only library materials of the specified type. Otherwise, library materials of all types are included. Each **Material** object is populated with the library material id, title, and type. The list of library materials should be sorted alphabetically by library material type and subsorted by library material title. If there are no library materials available for the specified title or partial title of the specified type, the method simply returns an empty list.

`public Material getMaterial(String materialId) throws SQLException` (see interface description)

The method will search for the library material with the specified material id in the **Material** table and, if found, will return a **Material** object populated with the library material id, title, and type. If a library material for the specified material id was not found in the **Material** table, then the method simply returns **null**.

The `AccountOwnerData` Class

Methods

`public AccountOwner createLibraryAccount(String firstName, String lastName, String username, String password) throws SQLException` (see interface description)

This method inserts a new library account owner into the **AccountOwner** table with the specified first name, last name, username, and password. The library account owner id is generated (identity column). The method then creates a new **AccountOwner** object populated with the generated library account owner id, and the specified first name and last name and returns it.

`public AccountOwner getAccountOwner(int id) throws SQLException` (see interface description)

The method will search for the library account owner with the specified library account owner id in the **AccountOwner** table and, if found, the method will return a **AccountOwner** object populated with the library account owner id, first name, and last name. If a library account owner for the specified library account owner id was not found in the **AccountOwner** table, then the method simply returns **null**.

`public AccountOwner validateAccountOwner(String username, String password) throws SQLException` (see interface description)

The method will search for the library account owner with the specified username and password in the **AccountOwner** table and, if found, the method will return a **AccountOwner** object populated with the library account owner id, first name, and last name. If a library account owner for the specified username and password was not found in the **AccountOwner** table, then the method simply returns **null**.

The `AccountAccessFilter` Class

The library account owner is considered logged in if there is a session-scoped attribute containing a **AccountOwner** object representing the logged in library account owner. The session-scoped attribute is

created by the **LibraryLoginControl**. If the session-scoped attribute for the current library account owner does not exist and the requested page is not the library login page or the new library account page, then the filter must redirect to the **librarylogin.jspx** page. Otherwise the filter simply continues along the filter chain.

The `LibraryLoginControl` Class

Page Association

The **LibraryLoginControl** class should be associated with the **librarylogin.jspx** page.

Methods

```
String doLogic(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
```

The **LibraryLoginControl** class should be modified so that the method should first try to retrieve the current **AccountOwner** object from the session. If an **AccountOwner** object is found in the session-scoped attribute, then the library account owner is already logged in. In this case, the method should redirect to the context root. If there is no library account owner object in the session scope, then the library account owner is not logged in yet. In this case, the method should get **username** and **password** from HTML form parameters and use them to validate the AccountOwner and, if the validation is successful, the method should store the **AccountOwner** object returned from the validation into a session-scoped attribute. The method should then redirect to the context root path.

If, on the other hand, the library account owner was not successfully validated, then the method should set a request-scoped attribute (to be used by the **librarylogin.jspx** page) indicating that the login was unsuccessful and continue along the filter chain. If the form method is not **POST**, then the method should simply continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

The `NewLibraryAccountControl` Class

Page Association

The **NewLibraryAccountControl** class should be associated with the **newlibraryaccount.jspx** page.

Methods

```
String doLogic(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
```

If the request method is POST and the non library account owner submitted using the create new library account submit button, the method should make the following checks:

- Get the first name from an HTML form parameter and make sure the first name is not null or empty. If the first name is valid, then it should be saved to a request attribute. If the first name is not valid, a request-scoped attribute should be set to indicate that.
- Get the last name from an HTML form parameter and make sure the last name is not null or empty. If the last name is valid, then it should be saved to a request attribute. If the last name

is not valid, a request-scoped attribute should be set to indicate that.
- Get the username from an HTML form parameter and make sure the username is not null or empty. If the username is valid, then it should be saved to a request attribute. If the username is not valid, a request-scoped attribute should be set to indicate that.
- Get the password from an HTML form parameter and make sure the password is not null or empty. The password should not be saved to a request attribute. If the password is not valid, a request-scoped attribute should be set to indicate that.

For each parameter value retrieved and sent back to the page in request-scoped attributes, be sure to **StringHelper.XmlEscape** the text first. If all of the above checks are successful, then the method should check that the password and the confirm password are the same. But if the password and confirm password do not match, the method should set a request-scoped attribute to indicate this. If there were no validation errors, the method should create a new library account and store the **AccountOwner** object returned from the account creation into a session-scoped attribute. Then the method should redirect to the context root. Otherwise the method should continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

The `ShowMaterialReservationsControl` Class

Page Association

The **ShowMaterialReservationsControl** class should be associated with the **showmaterialreservations.jspx** page.

Methods

```
String doLogic(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
```

The **ShowMaterialReservationsControl** class should be modified so that the list of library material reservations is retrieved for the current library account owner (saved in a session-scoped attribute by the **LibraryLoginControl** class).

librarylogin.jspx Page Changes

Modify the **librarylogin.jspx** page so that if the **LibraryLoginControl** class has saved a request attribute to indicate that the login was unsuccessful, the page should display the login error message.

newlibraryaccount.jspx Page Changes

The **newlibraryaccount.jspx** page must now retrieve request-scoped attributes indicating whether or not there were errors when specifying the new library account owner information and, if so, display an appropirate message indicating the error (first name not valid, last name not valid, username not valid, password not valid, and/or password and confirm password do not match). Depending on the situation, the page may display none, one, or more than one of the above messages.

showmaterialreservations.jspx Page Changes

If the library account owner has no library material reservations, the **showmaterialreservations.jspx** page must display a message indicating this. Otherwise the page must list the library material title and type, and expiration date for all library material reservations. The expiration date must be formatted as a date value using JSTL, showing day, month, and year. You do not need to display the time portion. Each of the cancel library material reservation forms must include a hidden form input for the material id (retrieved from the corresponding library material reservation object).

Group B Components

Members of this group are responsible for updating the classes **MaterialReservationData** to store and retrieve data from the database. The updated MVC **Control** classes for this group are also specified. The class updates are described below.

The `MaterialReservationData` Class

Methods

`public void cancelMaterialReservation(String materialId) throws SQLException` (see interface description)

This method deletes the library material reservation in the **MaterialReservation** table that has the specified material id. If a library material reservation with the specified material id is not found in the **MaterialReservation** table, the method will simply do nothing (no error is generated).

`public void deleteExpiredMaterialReservations() throws SQLException` (see interface description)

This method deletes all library material reservations in the **MaterialReservation** table where the expiration date & time is *before* the current date & time (specified as a **java.sql.Timestamp** type).

`public MaterialReservation getMaterialReservation(String materialId) throws SQLException` (see interface description)

The method will search for the library material reservation with the specified material id in the **MaterialReservation** table and, if found, will return a **MaterialReservation** object populated with the library material id, title, type, and expiration date & time (as a **java.sql.Timestamp** type) of the library material reservation. If a library material reservation for the specified material id was not found in the **MaterialReservation** table, then the method simply returns **null**.

`public List<MaterialReservation> getMaterialReservations(int accountOwnerId) throws SQLException` (see interface description)

Returns a list of **MaterialReservation** objects representing all of the library material reservations in the database for the library account owner with the specified library account owner id. Each **MaterialReservation** object is populated with the library material id, title, type, and expiration date & time (as a **java.sql.Timestamp** type) of the library material reservation. The list of library material reservations should be sorted by type and subsorted by title. If there are no library material reservations for the specified library account owner, the method simply returns an empty list.

`public boolean isMaterialAvailable(String materialId) throws SQLException` (see interface description)

Returns a boolean value indicating whether or not the library material with the specified library material id is currently available. The library material is available as long as it is not currently on reserve.

`public MaterialReservation reserveMaterial(int accountOwnerId, String materialId) throws SQLException` (see interface description)

This method inserts a new library material reservation into the **MaterialReservation** table with the specified library account owner id, library material id, and expiration date & time (7 days from now) — Be sure to send the expiration date to the database as a **java.sql.Timestamp**. The method then creates a new **MaterialReservation** object populated with the library material id, title, type, and the expiration date & time and returns it.

The `LimaraContextListener` Class

This class implements the **ServletContextListener** interface. When the servlet context is being initialized, the context listener acquires a **DataSource** object (e.g., using resource injection) for the library material reservation database and sets it as the data source to be used by the **DataSourceFactory** class. When the servlet context is being destroyed, the context listener sets the data source used by the **DataSourceFactory** class to **null**. Don't forget to add a **<listener>** entry into the **web.xml** file. Also, be sure to use **jdbc/limaradb** as the JNDI name of the database.

The `LimaraMenuFilter` Class

The LimaraMenuFilter class should no longer save the current library account owner into a request-scoped attribute (the current library account owner is saved in a session-scoped attribute by the **LibraryLoginControl** class). The filter should now get the number (how many) of library material reservations for the *current* library account owner and save that value into a request-scoped attribute to be used by the **limaramenu.jspx** page.

The `LogoutControl` Class

Page Association

The **LogoutControl** class should be associated with the fictitious **logout.jspx** page.

Methods

```
String doLogic(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
```

This method should end the current session and then redirect to the context root path.

The `ReserveMaterialsControl` Class

Page Association

The **ReserveMaterialsControl** class should be associated with the **reservematerials.jspx** page.

Methods

```
String doLogic(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
```

If the request method is POST, the method should make the following checks:

- Get the search library material type from an HTML form parameter and make sure the search library material type is not null or empty. If the search library material type is not null or empty, then it should be saved to a request attribute. If the search library material type is not valid (null or empty), a request-scoped attribute should be set to indicate that.
- Get the search library material title (or partial title) from an HTML form parameter and make sure the search library material title is not null or empty when the library material type has been set to **All**. If the library material type has been set to something other than **All**, then it is valid for the library material title (or partial title) to be null or empty. If the search library material title is not null or empty, then it should be saved to a request attribute. If the search library material title is not valid, a request-scoped attribute should be set to indicate that.

For each parameter value retrieved and sent back to the page in request-scoped attributes, be sure to **StringHelper.XmlEscape** the text first.

If there were no validation errors, the method should get a list of available library materials based on the search criteria and save the ist into a request-scoped attribute. If **All** was specified for the search library material type, use **null** instead. If the search library material title (or partial title) is empty string, use **null** instead. Additionally, if the library account owner submitted using the reserve material submit button, the method should:

- Get the material id from an HTML form parameter. If the material id is **null** or empty string, the method should throw a **BadRequestException**. Otherwise the method should get the **Material** object for the material id. If the material id is not valid (i.e., does not correspond to a library material), the method should throw a **BadRequestException**.
- If the material id is valid, the program should check that the specified library material is available. If it is not available, the method should set a request-scoped attribute to indicate the item is no longer available and then continue along the filter chain.
- If all of the above checks are successful and the library material is available, the method should reserve the library material using the material id specified. When reserving the library material, use the library account owner id for the current library account owner (saved in a session-scoped attribute by the **LibraryLoginControl** class) The method should then redirect to the context root.
- Since a library material can only be reserved by one library account owner, in the time between the check that the library material is available and the reservation of the library material, no other library account owner should be able to reserve the same library material. This can be accomplished using a synchronized block.

If there were validation errors, the method should continue along the filter chain. If the request method is not POST, the method should continue along the filter chain. If a **SQLException** occurs, the method should wrap the exception into a **ServletException** and throw it out of the method.

limaramenu.jspx Page Fragment Changes

The **limaramenu.jspx** page fragment must now display the library account owner's first and last name from a session attribute (saved by the **LibraryLoginControl** class) and incorporate them into the menu where required.

reservematerials.jspx Page Changes

The **reservematerials.jspx** page must now retrieve request-scoped attributes indicating whether or not there were errors when specifying the library material search criteria or when specifying the new library material reservation and, if so, display an appropirate message indicating the error (library material type not valid, library material title cannot be empty, library material is no longer available). Depending on the situation, the page may or may not display one of the above messages.

The page must also now retrieve the library material type from a request attribute (if it exists) and have that library material type selected in the drop down list when the page loads (hint: use JSTL to set the **selected="selected"** attribute for the <option> representing the current library material type).

If a list of available library materials has been provided (through a request scoped attribute), then the page must include the following:
If there are no available library materials (i.e., the list is empty), the **reservematerials.jspx** page must display a message indicating this. Otherwise the page must list the material title and type of all available library materials. Each of the reserve library material forms must include hidden form inputs for the material id as well as the search library material title (or partial title) and search library material type.

Back to Overview Stage 2